A cloud-based architecture for an affective recommender system of learning resources

Derick Leony, Abelardo Pardo, Hugo A. Parada G., Carlos Delgado Kloos
Department of Telematic Engineering
Universidad Carlos III de Madrid
Leganés, Madrid, Spain
Email: {dleony, abel, hparada, cdk}@it.uc3m.es

***Abstract*—One of the most common functionalities in cloud-based learning environments is the recommendation of learning resources. Many approaches have been proposed to deploy recommender systems into an educational environment. Currently, there is an increasing interest in including affective information into the process to generate the recommendations for the learner. In this paper, we propose a cloud-based architecture for a system that recommends learning resources according to the affective state of the learner. Furthermore, we provide the details of an implementation of the architecture along with a discussion on the advantages and disadvantages of the proposal.**

***Keywords*-cloud educational environment; learning resource recommendation; affective recommendation**

I. Introduction

An important characteristic of learning environments in the cloud is the personalization of the environment according to the learner needs, objectives and current situation. Thus, the learning environment can adapt its interface, content, and capabilities according to personal characteristics of the learner: current learning objectives, learning achievements, skills, preferences and affective state. In addition, contextual information is another input for personalization: current location, time of the day and available technology.

Recommender systems are among the instruments used to provide learning environments with personalization. Manouselis et al. [1] provide a review of the approaches followed so far to implement and deploy recommender systems in Technology Enhanced Learning (TEL). More recently, some approaches are including affective information of the learner, such as the case of the Semantic Affective Educational Recommender System proposed by Santos and Boticario [2].

In this paper, we present an improvement of the architecture presented in [3] of the Learning Resource Affective Recommender (LRAR). The most relevant change for the cloud context is the deployment of the recommending engine as an auto-scaling service, which allows to serve several clients with reasonable response time and performance. Thus, the recommender engine is transformed into a recommending service based on cloud technologies. The migration to the cloud allows us to improve the scalability of the recommender system, given the high level of computational power required by this kind of processing.

The rest of the paper is structured as follows: section III describes the proposal of a cloud-based architecture for an affective recommender system of learning resources, Section IV provides the details of an implementation of the architecture, and section V presents some points for discussion.

II. Related works

Most systems on the internet are currently supported on cloud computing. This trend also is observed in the educational arena. The interest of researchers on cloud in learning technologies has increased in last few years. In this section we present some of the most recent works regarding the application of cloud technology into the learning domain.

Cloud technologies are a set of easily accessible and virtualized resources that can be dynamically adapted allowing an optimum resource utilization [4]. A cloud technology allows users access onto different services on internet through diverse devices. It also provides service providers with several advantages, such as availability, integration of multiple services, flexibility and scalability [5]. This makes cloud technology an attractive option for deploying applications that require a high level of computational power. Hence cloud technology has taken into account to support services that offer educational resources to academic communities.

In this sense, in [6] Mikroyannidis states that the cloud offers a lot of services for building adaptive and customisable Cloud Learning Environments (CLE). He also explains how CLEs extend the borders of the learning environments beyond of educational organization. Additionally this work proposes a learning scenario based on the use of cloud learning services. Thus, to take advantage of these features educational institutions are also moving towards providing their services by using cloud technologies. However, in the educational domain, services are scarcely adapted and offered in cloud.

Several approaches of cloud architectures promote improvements to services in the e-learning area by using cloud technologies. Thus, they try to overcome challenges faced

by educational institutions. In [7] Masud and Huang propose an e-learning cloud architecture to allow the migration of e-learning systems from schools to a cloud computing infrastructure. They describe an e-learning cloud architecture made up of five layers: infrastructure, software resource, resource management, service and application. This proposal describes a general architecture for e-learning; nevertheless it does not focus on how to implement an e-learning service in a cloud architecture.

As CLE appears as a set of available tools on the internet that allows ubiquitous access to an academic community, it is evident that the existing of Personal Learning Environments in the cloud are in an early stage of its developing. Currently the CLE has dealt in offering an environment that allows students and teachers easy access to different tools for producing and consuming academic content. A related work is also presented by Al-Zoube in [8], where he proposes a cloud computing based solution for building a virtual Personal Learning Environment (PLE). This proposal consists of allowing the learner access to different tools offered on internet such as iGoogle, Google docs, YouTube, etc. however such as Stein et al. state in [9] public clouds generally meet the common base of user requests, but they may not be designed to meet educational needs. In addition, today learners are demanding specialized learning services in order to improve the learning results. Hence, the educational domain requires design and deploy services in order to built a true educational Cloud.

Following the above approach, in [10] Madan et al. present a cloud-based learning service model. This proposal describes comprehensively the cloud computing services as a key aspect of cloud computing model. The authors focus on services and available models to be deployed into cloud architecture. They claim that institutions should use the existing cloud infrastructure offered by companies such Google, Amazon and others. Then educational institutions should focus on defining the cloud service layer to implement it into the cloud architecture.

There is a known necessity of building a CLE based on specialized services rather than the traditional tools found in PLEs. However as the necessities of learning resources and services for learners and teachers are variable, we must offer a service to adapt the PLE in the cloud according to these necessities. In other words, CLE needs a recommender system to fill this gap. Recommender systems have been extensively deployed, however few systems operate in the education arena [11]. Then in this work we propose a cloud-based architecture for a system to recommend learning resources according to the affective state of the learner. Thus learners will be able to adapt their PLE and CLE. Additionally we provide details of the architecture implementation of this specialized service.

III. Description of the architecture

The purpose of the proposed architecture is to deliver a set of learning resources meta-data following a Software as a Service approach. The architecture is composed by two layers: a service layer that executes the storage and recommendation tasks, and a client layer embedded in a learning environment. The details of each layer and their communication are described as follows.

A. Service layer

The service layer is in charge of receiving petitions from several clients and doing the requested tasks. The available tasks are to update the affective information of a learner, to update the information of a learning resource, and to recommend learning resources for a given learner. The first two tasks represent an administration interface for the management of learners' affective states and learner resources. The third task is the main one in the service and also the one that consumes the most resources.

The service layer includes two storage elements to keep the information of the learning resources and the learners' affective states. The storage of learning resources only includes their meta-data and not their content. This decision relies on the fact that the recommendation service is not meant to act as a repository of learning resources but just as a referrer. The format of the database can be any usual specification such as Learning Object Meta-data (LOM), but this is decided by the implementation of the architecture.

The storage for learners' affective state is updated by requests from the client layer. The service is ready to create a new learner profile which consists of the learner identifier, current affective state and the record of the used resources and the affective state presented when using those resources. The format used to define the affective state is decided by the implementation of the architecture as well. The specification EmotionML should be strongly considered because, although still being a W3C Candidate Recommendation, it allows flexible and complete definitions of affective states.

Besides the storage elements, the service layer has a recommendation engine cluster. The engine is designed as a cluster because the task to generate recommendations is the most expensive in terms of computational resources, which makes it the critical process to scale. The cluster contains as many instances of a recommending engine node as needed to support the service demand at the moment. Each node is in charge of analyzing the resources, the affective states of a learner and generate assign a relevance score to each resource not yet seen by the learner.

The recommendation process within a node follows the method known as user-based collaborative filtering. In this approach, when a recommendation has to be done for a given learner, it first finds a set of the learner's *neighbors*, learners with similar patterns of access to resources. The level of similarity is to identify the neighbors of a learner

is defined by a similarity function. In the case of the affective recommendation, the similarity of two learners is proportional to the amount of resources accessed by the learners when indicating the same affective state.

As the recommendations must take into account the affective state of the learner, the collaborative filtering process had to be extended to include that contextual information. The set of resources available for recommendation are a combination of the learning resources with the affective states. Thus,

$$R = L \times A$$

where R is the set of recommendable resources, L is the complete set of resources meta-data and A is the complete set of affective states. The recommendable resources are a tuple of a resource and an affective state.

After the learner's neighborhood is defined, learning resources are sorted based on how relevant they have been within that neighborhood of users. The resulting list must be filtered because it contains recommended tuples *(resource, affective state)* and the affective state might be any stored one. Thus, the list of recommended resources are only those that appear in a tuple where the affective state is the same one the learner presents at the moment.

B. Client layer

In the presented architecture, the client layer is represented by the learning environment that includes recommendations to provide adaptation. The inclusion of recommendations is done by an embedded element deployed within the learning environment. The embedded element communicates with the service layer to send and request information related to the recommendation based on affective states.

The embedded element sends update information such as the learner identifier and any change of her affective state. The element also informs when a learner accesses a learning resource. Optionally, the embedded element can also be in charge of detecting the learning state of the learner and updates the affective state database. Other optional information to send is any action done by the learner; this allows to keep track of the learner actions for further analysis.

There can be two moments when the embedded element requests information from the service layer. First, when accessing the learning environment the first time the client requests the last known affective state and the last recommended resources. The second moment is right after the affective status of the learner is updated. This is because a modification of the affective status implies a new set of recommendations for the learner, so the embedded element requests the list of recommended resources. After fetched, the list is displayed showing the title and description of each resource. Then, the learner is able to select a resource based on its description or on the relevance score given by the recommendation engine. Figure 1, shows a diagram of the proposed architecture. The layers are displayed from bottom to top.

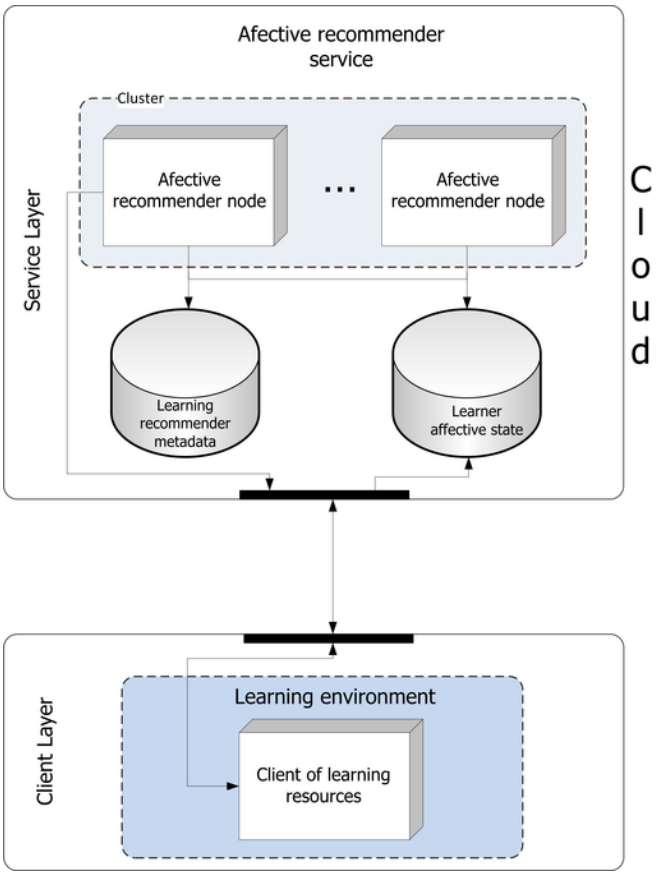

Figure 1. Architecture of affective recommender system based on the cloud.

IV. Explanation of the Implementation

The environment where we have implemented the affective recommender service is Amazon Elastic Computing Cloud (EC2), which is part of Amazon Web Services (AWS). EC2 allows us to define an image that acts as a blueprint to generate several instances of a computer with the same software configuration. Each one of these instances is what in the definition of the architecture we have called a node.

In our implementation, the storage element for learner resources meta-data is implemented as a database deployed in the engine MySQL. The same database implements the learner affective state storage element. Since the data might be accessed from many nodes of the cluster, the database engine is installed in an independent node, not meant to be part of the recommendation cluster. In order to manage the information stored in the database, the database node implements a RESTful web services, while the technology supporting the web application is J2EE.

The recommendation engine is developed on top of Apache Mahout machine learning engine. Mahout provides a set of libraries to implement machine learning models such as collaborative filtering, recommender systems, clustering, pattern mining and classifiers. Mahout is implemented in Java and this allows a straightforward integration with a web application developed with J2EE. As explained in [12], Mahout is conceived to be scalable through the framework for distributed processing Apache Hadoop, which allows the definition of clusters of computers with computational and storage capabilities.

The algorithms for collaborative filtering use the map/reduce paradigm. This paradigm consists in two processes that can be parallelized and distributed among several computers to increase their speed. The *map* process generates a sequence of pairs where usually the first element is an entity identifier and the second element is an entity characteristic that will be needed in a further computation. The *reduce* process receives the pairs generated by the map process and computes an incremental value associated with the entity. For example, the first step to identify the neighbors of a learner is to identify the most frequent occurrences of a pair *(affective state, learner resource)* for each user. In this case, the map process returns a pair with the syntax *(user identifier, (affective state, learner resource))*. Thus, the entity to identify is the user and the other item to include is the pair or affective state an resource. The reduce process receives the same pair and create an array for each received user. The elements of the array are complex structures with the syntax *((affective state, learner resource), count)*, so that by sorting the array in descendant order by the second element we obtain the top occurrences of pairs for the given learner.

The advantage of using the map/reduce paradigm is that both process can be done in parallel and in several computers. This allows the implementation to scale by just creating a new node with the same characteristics of a previous recommender node. Hadoop keeps control of the nodes that are available in the cluster to perform computational and storage tasks. Thus, we are provided with a simple way to auto-adjusting the size of the recommender cluster by just adding or removing nodes according to the service demand.

For the implementation of the client layer element, a widget has been developed as a proof-of-concept of a tool that interacts with the affective recommender service. The widget has been developed using the Software Development Kit (SDK) provided by ROLE Project [13]. ROLE aims to provide the learner with a framework to build her Personalized Learning Environment. The widget is implemented in JavaScript and HTML, and it follows the OpenSocial Gadget specification.

The widget interface has two functional sections represented by the tabs. Resources, the main tab, allows the learner to state her affective state from a static list provided by the recommendation service. Currently, the list is based on the affective states used by D'Mello et al. in [14]; these include frustrated, confused, bored, enthusiastic, motivated and the normal state, meaning that there is no relevant affective state at the moment.

Resources tab also presents a list of learning resources ordered by relevance for the learner in her current state. Thus, once the learner submits a change on her affective state, the widget send a recommendation request to the affective recommender service. When the response is received, the client analyzes the list of resources and embed their information as the list of recommended resources.

Second section is the *Profile* tab, where a time-line of the affective states reported by the learner is embedded. Its objective is to provide the learner with a visualization of her emotional changes during the learning activity being performed. The log of affective states is also provided by the learning resource service.

Finally, the Settings tab allows the learner to set her learning objectives. These might be changed during the learning activity, which also triggers a change of the learning resources that are recommended. Figure 2 presents a screen capture of the widget deployed in ROLE environment, with emphasis on the resources recommended to a frustrated learner.

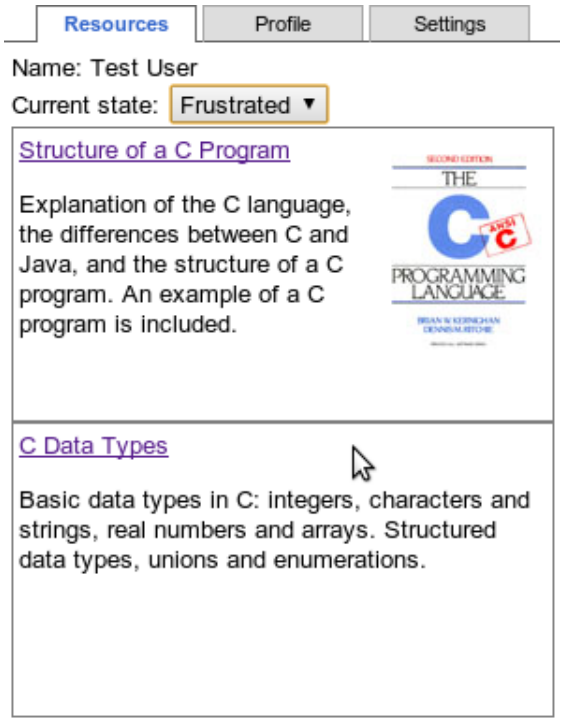

Figure 2. Implementation of the resource visualizer in the ROLE PLE.

A use case that exemplifies the use of the widget is the following. Alice, a university student whose major is

Computer Science, is trying to complete a C programming task that she was assigned as homework. She starts working highly motivated on the initial details of the program and she reflects be selecting the *Motivated* option among the affective states available in the widget. The widget communicates the affective state to the recommender service and this returns a list of resources suitable for Alice, such as C programming references and the user's manual of the compilation tool. Alice finds the resources helpful and uses them to write her code more quickly. As Alice advances she realizes that the task is not as easy as she first thought and that she might even encounter some programming errors that she was not expecting, this causes her affective state to change. When she find that she cannot fix a compilation error she starts feeling frustrated. Thus, she updates her affective state to *Frustrated*. Again, the widget communicates the new affective state to the recommender service, obtains the list of suitable resources and displays them as part of the widget content. The new list of recommended resources includes basic programming concepts and common programming errors with their respective solutions (see Figure 2). Alice accesses the resources and solves the error of her program. After seeing that the widget helps her along the task, Alice recovers her positive mood and continues her work to finish the homework.

V. Discussion

Throughout this paper we have pointed the advantages of a cloud-based affective recommender system of learning resources. We have emphasized that the main benefit of using cloud technologies to deploy an affective recommender system is the gain of scalability. Nevertheless, there are some issues that must be taken into account as possible disadvantages of the architecture such as approaches to caching and privacy issues.

The first issue is the difficulty to cache results in the presented architecture. Given that in a cloud recommendation service the computational workload is distributed among the nodes of the cluster, one node cannot cache the recommendation calculate in another node. This issue can be addressed either at the client layer, where the client itself would be in charge of caching the information of the recommended resources. Another possible approach to solve this issue is through the inclusion of a middle layer that caches and dispatches the recommendations; this layer would be between the recommendation cluster and the interface of the service.

Another concern is related to privacy issues, since the affective state of the learner is stored in a database accessible from several recommender nodes. The key in this issue is that the recommending nodes are the only elements with permission to access the affective state information. Furthermore, the recommender service does not require to store information that identifies the learner immediately, such as the full name or email. Instead, the service can use a hashed identifier of the learner and that would not interfere with the process of recommending learning resources.

Future work consists on evaluating the performance of the implementation presented in the article. The evaluation objective is to analyze the improvement on the response time of a recommendation request to the server. Part of this work includes to analyze the correlation between the service performance and the amount of recommender nodes in the cluster; this would lead to a set of guidelines for deploying the recommender service in a real learning scenario.

Another line of work consists in the development of plug-ins to include sensors as a method to populate the affective state database. Specifically we are working with sensors for galvanic skin response and the recognition of face gestures through a video camera. These sensors have been proven to detect affective states with accuracy [15] and thus might be improve the recommendation process. They would also allow the learner to focus on the retrieval and use of resources rather than constantly informing her current affective state. On the same track, it is also intended to improve the interface for the learner to provide her affective state. Several approaches will be taken in order to obtain contextual information about what provoked a given emotion in the learner and how did the recommendation of resources affect her affective state.

Acknowledgment

Work partially funded by the EEE project, "Plan Nacional de I+D+I TIN2011-28308-C03-01", the "Emadrid: Investigación y desarrollo de tecnologías para el e-learning en la Comunidad de Madrid" project (S2009/TIC-1650), and "Consejo Social - Universidad Carlos III de Madrid".

References

[1] N. Manouselis, H. Drachsler, K. Verbert, and E. Duval, "Recommender systems for learning," *SpringerBriefs in Computer Science*, 2012.

[2] O. Santos and J. Boticario, "Affective issues in semantic educational recommender systems," in *Proceedings of the 2nd Workshop on Recommender Systems for Technology Enhanced Learning (RecSysTEL 2012). Manouselis, N., Draschler, H., Verber, K., and Santos, OC (Eds.). Published by CEUR Workshop Proceedings*, 2012, pp. 71–82.

[3] D. Leony, A. Pardo, H. A. Parada G., and C. Delgado Kloos, "A widget to recommend learning resources based on the learner affective state," in *Proceedings of the 3rd Workshop on Motivational and Affective Aspects of Technology Enhanced Learning (MATEL 2012).*, In process.

[4] L. M. Vaquero, L. Rodero-merino, J. Caceres, and M. Lindner, "A Break in the Clouds: Towards a Cloud Definition," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50–55, 2009.

[5] N. Leavitt, "Is cloud computing really ready for prime time?" *Computer*, vol. 42, no. 1, pp. 15–20, jan 2009.

[6] M. Alexander, "A Semantic framework for cloud learning environments," *In: Chao, Lee ed. Cloud Computing for Teaching and Learning: Strategies and Implementation. Hershey, PA: IGI Global*, pp. 17–31, Apr. 2012.

[7] X. Huang, "An E-learning System Architecture based on Cloud Computing," *International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE)*, vol. 2, pp. 74–78, 2012.

[8] M. Al-zoube, "E-Learning on the Cloud," *International Arab Journal of e-Technology*, vol. 1, no. 2, pp. 58–64, 2009.

[9] S. Stein, J. Ware, J. Laboy, and H. E. Schaffer, "Improving K-12 pedagogy via a Cloud designed for education," *International Journal of Information Management*, http://dx.doi.org/10.1016/j.ijinfomgt.2012.07.009 2012.

[10] A. A. Deepanshu Madan, Suneet Kumar, "E-learning based on Cloud Computing," *International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE)*, vol. 2, no. 2, 2012.

[11] K. Verbert, N. Manouselis, X. Ochoa, M. Wolpers, H. Drachsler, I. Bosnic, and E. Duval, "Context-aware Recommender Systems for Learning: a Survey and Future Challenges," *To appear in IEE Transactions on Learning Technologies*, 2012.

[12] S. Owen, R. Anil, T. Dunning, and E. Friedman, *Mahout in action*. Manning Publications Co., 2011.

[13] R. Consortium, "ROLE Project," http://www.role-project.eu/, 2009-2012, last visited September 2012.

[14] S. D'Mello, A. Graesser, and R. Picard, "Toward an affect-sensitive autotutor," *Intelligent Systems, IEEE*, vol. 22, no. 4, pp. 53–61, 2007.

[15] R. Picard, *Affective computing*. The MIT Press, 2000.